Chapter 11: Encrypted vs. Unencrypted Connections

In this chapter, we provide guidance on determining whether your connection is encrypted, and ensuring that you open an encrypted connection, as needed.

To comply with Fermilab policy, you only strictly need an encrypted network connection when you type your Kerberos password. And to further comply with policy, you should type your Kerberos password over the network extremely rarely, if at all!

11.1 How do you know if your connection is encrypted?

When you're connecting to a strengthened machine over the network, it's very important to know if your connection is encrypted. If it is, you can reasonably safely run Kerberos commands that require input of your Kerberos password (see note above). If your connection is not encrypted, you must not type your Kerberos password, since it would be transmitted in the clear. Notice that there are lots of bombs in this chapter!

If you have a chain of multiple connections (e.g., machine1 to machine2, machine2 to machine3, and so on), and if only one connection is unencrypted, then your connection as a whole is **unencrypted**. Do not type your Kerberos password in this case!

Now, we'll discuss the individual connections ...

11.1.1 Connecting from Kerberized UNIX/Linux Desk-

tops

SSH

If you connect via Kerberized ssh, verify your ssh client configuration to make sure it initiates encrypted sessions. This will vary depending on the ssh client. If you're not sure, use the command with the **`-c`** option as follows:

```
% ssh -c 3des <host>
```

or other argument to `-c` (except for **`none`**).

Other Kerberized Connection Program (e.g., telnet)

Your connection is encrypted if you are connected via one of the Kerberized programs with the "encryption on" flag set. The program generally tells you. For example, for telnet, you can tell if the default is set for encryption by typing the escape character (default is **CTRL-]**, it can be reset with **`-e`** flag), and entering **`status`**. Encryption information should be listed.

For any Kerberized connection program, you can always check the default setting in the `[appdefaults]` section of your `/etc/krb5.conf` file. Look for `encrypt=true` for the program you're using. If encryption is not on by default, use the encryption flag, e.g.,:

```
% rsh -x <host>
```

```
% telnet -x <host>
```

where **`rsh`** and **`telnet`** reside in `/usr/krb5/bin`. Reference Chapter 13: *Network Programs Available on Kerberized Machines* for command syntax.

If you connected in a different way, or if you're not sure, then **assume that the connection is not safe**, log out and log in again as shown here.

11.1.2 Connecting over a CRYPTOCard ssh Session

Verify your ssh client configuration to make sure it initiates encrypted sessions. This will vary depending on the ssh client. If you're not sure, use the command with the **`-c`** option as follows:

```
% ssh -c 3des <host>
```

or other argument to `-c` (except for **`none`**).

11.1.3 Connecting over a CRYPTOCard telnet Session

CRYPTOCard telnet connections are **unencrypted**, and it's **never safe** to issue your Kerberos password. See section 11.2 *If it's unencrypted, what do I do when I need to reauthenticate?*.

11.1.4 Connecting over a CRYPTOCard ftp Session

CRYPTOCard ftp connections are **unencrypted**.

11.1.5 Connecting from an X Terminal

The connection from an X terminal to a host is **never encrypted**, so you must **never** issue your Kerberos password from an X terminal, no matter how secure the connections are beyond that point.

11.1.6 Connecting from a PC Running Windows

Helpful hint: look for the locked lock symbol in your session window to ensure the connection is encrypted!

With ssh

This will vary depending on the ssh client. Verify your client configuration to make sure it initiates encrypted sessions.

With WRQ® telnet client

WRQ® Reflection Security Components v8.0.0 supports ticket forwarding to the remote host, so you shouldn't need to run any commands on the remote system that require password entry. Therefore you may not need an encrypted connection (see section Chapter 19: *Configuring WRQ® Reflection telnet Connections*). If you need to type your password on the remote host for any reason, then you do need an encrypted connection. Make sure that the **WRQ® Reflection** telnet client is configured as described in section 19.8 *Configuring WRQ® Reflection telnet Connections*.:

If you've installed **WRQ® Reflection X**, you can opt to connect to a host directly from the **X CLIENT MANAGER** window, *but* it **does not provide encrypted connections**. If you will need credentials on the host, go through a normal **telnet** connection. **Do not `kinit` from an X window!**

With MIT Kerberos and Exceed 7.0 telnet client

Exceed 7.0 supports ticket forwarding to the remote host, so you shouldn't need to run any commands on the remote system that require password entry. Therefore you may not need an encrypted connection. If you need to type your password on the remote host for any reason, then you do need an encrypted connection. To enable encryption, configure your Kerberized Exceed 7.0 telnet connections as described in section 21.5 *Configuring the Exceed 7 Telnet Application*.

11.1.7 Macintosh: MIT Kerberos and BetterTelnet

In section 23.5 *Configuring Telnet* pay attention to item (3). To summarize:

Invoke **BetterTelnet**. On the **FAVORITES** menu, choose **EDIT FAVORITES**. On the pop-up screen, click **NEW** to create a new configuration or edit an existing one. Change to the **SECURITY** tab, check `Kerberos authentication` and `Kerberos encryption`. Click **OK** to save the configuration.

11.2 If it's unencrypted, what do I do when I need to reauthenticate?

One option for updating tickets on remote sessions is to use the `k5push` script documented in section 9.2.6 *Update Tickets on Remote Terminal Sessions*.

For portal mode connections, a script is provided with the Fermi **kerberos** product as of version v1_2, that safely reauthenticates you on a Kerberized host using your CRYPTOCard over an unencrypted connection. The process exploits the portal mode feature that telnet with a CRYPTOCard always gets you a new key. The script is found at `/usr/krb5/bin/new-portal-ticket` (the script content is provided at the end of this section). Here's how it works:

From your X terminal or unstrengthened machine, you run telnet to a Kerberized machine and use your CRYPTOCard to authenticate. You get logged in on, say, `/dev/ttyp3`, with Kerberos tickets cached in `/tmp/krb5cc_ttyp3`. Now your ticket expires. Still logged into the Kerberized node, you log into the same machine again but using the nonKerberized telnet:

```
% /usr/bin/telnet localhost
```

The machine responds in portal mode, you use your CRYPTOCard, and you're now logged in on `/dev/ttyp4`, for example, with a new `/tmp/krb5cc_ttyp4` file that has a new cached Kerberos ticket.

So now you have two Kerberos cache files, and you're logged into the machine twice. One cache file (`/tmp/krb5cc_ttyp3`) has an old expired ticket in it, and the other (`/tmp/krb5cc_ttpy4`) has a fresh, new, usable ticket.

Next, copy your fresh `/tmp/krb5cc_ttyp4` file onto `/tmp/krb5cc_ttyp3` (both cache files live on the destination machine, so you're doing a safe, local file copy), run **`kdestroy`** (which removes `/tmp/krb5cc_ttyp4`), and log out once. Now you're back at `ttyp3`, with a fresh new kerberos ticket in `/tmp/krb5cc_ttyp3`, and you can continue doing whatever you were doing when your ticket expired.

Script Contents

```
#!/bin/sh

# get uid
eval `id | sed -e 's/(.*//'`

# figure ticket cache
if [ "x$KRB5CCNAME" = x ]
then
    krb5file=/tmp/krb5cc_$uid
else
    krb5file=`echo $KRB5CCNAME | sed -e sxFILE:xx`
fi

(
   read line
   echo $line
   sleep 1000 &
   pid=$!
   echo 'rkrb5file=`echo $KRB5CCNAME | sed -e sxFILE:xx`'
   echo "cp \$rkrb5file $krb5file"
   echo "kdestroy"
   echo "echo xyzzy $pid xyzzy"
   echo "exit"
   wait $pid
) | (
    /usr/krb5/bin/telnet localhost
) |
    while read line
    do
        set : $line
        case $2 in
        Press)
    printf "$line\n"
    printf "Enter the displayed response: "
    ;;
        xyzzy)
    kill $3
    ;;
        esac
    done
```